

Maude-HCS: Model Checking the Undetectability-Performance Tradeoffs of Hidden Communication Systems

Joud Khoury
RTX BBN Technologies
Cambridge, MA, USA
joud.khoury@rtx.com

Minyoung Kim
SRI International
Menlo Park, CA, USA
minyoung.kim@sri.com

Christophe Merlin
RTX BBN Technologies
Cambridge, MA, USA
christophe.merlin@rtx.com

José Meseguer
U. of Illinois at Urbana-Champaign
Urbana, Illinois, USA
meseguer@illinois.edu

Zachary Ratliff
Harvard University
Cambridge, MA, USA
zacharyratliff@g.harvard.edu

Carolyn Talcott
SRI International
Menlo Park, CA, USA
carolyn.talcott@sri.com

Abstract

Hidden communication systems (HCS) embed covert messages within ordinary network activity to hide the presence of communication. In practice, the *undetectability* of an HCS is typically evaluated using ad hoc traffic statistics or specific detectors, making security claims tightly coupled to experimental setups and implicit adversarial assumptions. In this work, we formalize undetectability as the statistical indistinguishability of observable execution traces under two deployments: a baseline system without hidden communication and an HCS deployment carrying covert traffic. Undetectability is expressed as a bound on a quantitative measure of distance between the trace distributions induced by these two executions.

We develop Maude-HCS, an executable modeling and analysis framework that provides a principled and executable foundation for reasoning about undetectability-performance tradeoffs in complex HCS designs. Maude-HCS allows designers to specify protocol behavior, adversary observables, and environmental assumptions, and to generate Monte Carlo samples from the induced trace distributions. We demonstrate that Maude-HCS can be used to audit claims of undetectability by estimating the true and false positive rates of a statistical test and converting these estimates into lower bounds on undetectability measures such as KL divergence. This enables systematic evaluation of detectability and its tradeoffs with performance under explicitly stated modeling assumptions.

Finally, we evaluate Maude-HCS on proof-of-concept tunneling-based HCS instantiations and validate model predictions against measurements from a physical testbed. For passive adversaries observing timing and traffic statistics, we quantify how undetectability and performance vary with protocol configuration, background traffic, and network loss, and demonstrate strong semantic alignment between model-based guarantees and empirical results.

Keywords

hidden communications, detectability, networks, model checking, quantitative properties

1 Introduction

Hidden communication systems (HCS) embed covert messages into ordinary network activity, with the goal of concealing not only the content of communication but its very existence. Such systems are critical for both the internet freedom and the national security communities [18–20]. For example, they enable journalists (and warfighters) to covertly coordinate and circumvent online censorship in authoritarian and/or adversarial environments. The use of these systems carries significant risk, where *detection of “fact of communication”* can result in loss of life [21].

A broad range of hidden communication systems have been proposed and/or deployed. The techniques used by these systems may be broadly classified into protocol tunneling (e.g., [7, 8, 36, 52, 55, 61]), mimicry (e.g., [22, 23, 46]), and/or obfuscation (e.g., [60, 62, 63]); reflection/refraction (e.g., [25, 30, 47, 59]); cloud fronting and proxying (e.g., [27, 64]); and steganography (e.g., [15, 37]). An important property common to these systems is how they balance *performance* with risk of *detection*. Different techniques occupy different areas in the undetectability-performance solution space. Both performance (e.g., the time it takes to send a message of a certain size) and risk of detection by the adversary are of paramount importance to the HCS user, whose needs are generally driven by mission requirements (e.g., a fixed window of time to deliver a message of a certain size at an acceptable level of risk).

Although hidden communication systems have been extensively studied in the literature, no general and scalable framework or tool exists today for formally modeling and reasoning about their performance and undetectability properties. Validating undetectability and performance properties of HCS designs today remains largely informal, relying on simulations and testbed experimentation of the HCS implementations. While this approach has been very valuable to date, it is rigid, error-prone, and takes significant time and effort. In practice, this lack of formal models of HCSs (including their implicit and explicit assumptions) and of a principled and modular approach for reasoning about their properties makes it hard to effectively explore the design and parameter space, to evolve the design, or to compare different designs.

On the other hand, state-of-the-art techniques have significantly advanced the theory and practice of formalizing distributed network system designs and reasoning about their security [6, 9, 11, 16, 39, 54] and performance properties [32, 33]. These formal approaches, however, have not been applied to the HCS domain, and

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit <https://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.



Proceedings on Privacy Enhancing Technologies 2026(4), 1006–1021

© 2026 Copyright held by the owner/author(s).

<https://doi.org/10.56553/popets-2026-0156>

they remain very tedious in practice requiring highly skilled experts. In particular, many design choices must be made to choose suitable formalisms, abstractions, analysis frameworks and properties, and scaling techniques.

We present Maude-HCS, *the first generalized toolchain for formally specifying and reasoning about hidden communication systems at real-world scales*.¹ The toolchain, comprised of a language and analysis toolkit, enables network designers to model and explore alternative HCS designs rapidly and effectively, and validates privacy-performance guarantees needed to trust the design — a necessary step for HCS users to ultimately trust the system, especially when operating in high threat environments.

Maude-HCS brings state-of-the-art formal tools and frameworks to the HCS domain. The basis of the toolchain is Maude’s expressive, extensible, and executable language based on rewriting logic [40]. Rewriting logic, with its real-time and probabilistic rewrite theories [2, 12, 14, 49], is ideally suited for formalizing distributed object-oriented systems such as HCSs. The formalism is timed and probabilistic, thus able to represent pertinent behaviors and properties of HCSs and adversaries hunting for them (e.g., stochastic message sizes, arrivals, delays, computation time, timers and timeouts, traffic and behavior models, and randomized algorithms). The formalism is also general and faithful, so it can express a wide range of hiding techniques and applications without compromising fidelity. The formalism is executable with high performance, enabling in-situ, fast, and automated formal analysis of HCSs.

To enable scalable and rigorous analysis, the Maude-HCS toolkit supports verification of precise quantitative privacy and performance properties, expressed in the expressive QuaTeX quantitative and probabilistic temporal logic [2, 3] using Statistical Model Checking (SMC) tools in Maude and QMaude [26, 53].² SMC reasons about precise quantitative properties while remaining scalable by estimating statistical quantities up to a configurable confidence level using Monte Carlo simulations [2, 3].

This work makes the following key contributions.

- **Formalizing undetectability over observable trace distributions.** We formalize *undetectability* as a quantitative indistinguishability condition between the distributions of observable execution traces induced by the ordinary system (without hidden communications) and the HCS (with hidden communication). Undetectability is expressed as a bound on a divergence measure between these trace distributions. This formulation makes explicit the dependence of undetectability on *observable* trace semantics, and provides a precise language for stating quantitative undetectability claims.
- **Modular semantically-aligned models and scalable analysis.** Maude-HCS integrates state of the art tools and techniques [33, 42, 44, 53] to reduce the modeling and analysis burden on HCS designers. We build on the state-of-the-art modular approach of Liu et al. [33] for formally specifying concurrent distributed systems with time and probabilities as generalized actor theories [43]. Designers specify non-deterministic models, which are automatically transformed

using sound theory transformations into probabilistic versions that are semantically aligned. We leverage a recent important extension [44] that allows specification of timed models as input to the transformation, to allow faithfully expressing inherent semantics in HCSs such as timers and timeouts. To further encourage generality and modularity of HCS specifications, we systematize different classes of HCSs as formal patterns [28, 42]. The tunneling pattern for instance embeds hidden data in legitimate cover protocol sources.³ The approach significantly reduces the burden on the designer for implementing HCS models so they can focus on modeling the novel aspects of their HCS design, while ensuring semantic consistency of the overall model, and thus increases trust in the analysis results.

- **Methodology for auditing undetectability claims.** We introduce a methodology for auditing divergence-based undetectability claims for concrete HCS designs using SMC. Given a protocol, a specified adversary observation model, and a claimed privacy parameter, Maude-HCS enables designers to sample observable traces under both ordinary and HCS executions and to compute adversary-relevant statistical quantities under each model. These measurements allow practitioners to assess whether the claimed divergence bound is supported under the specified assumptions and to explore how undetectability varies as system parameters (e.g., adversary detection logic or application performance) are modified. We demonstrate the predictive power of our approach by comparing model-based guarantees to actual experimental results on a controlled testbed, and showing strong alignment between the two.
- **Scalable auditing across parameterized deployment regimes.** We demonstrate that Maude-HCS enables systematic auditing of undetectability-performance tradeoffs by sweeping deployment and adversary parameters. Across multiple scenarios, we vary application rate, network loss, background traffic, and detector thresholds, and compute measures of undetectability for each operating point. This shows that the tool supports structured exploration of large design spaces rather than single-point evaluation.

The paper is organized as follows. Section 2 introduces relevant background material. Section 3 introduces our formal definition of undetectability over observable trace distributions. Section 4 describes our pipeline and methodology for modeling HCS systems and auditing their properties. Section 5 presents a thorough evaluation of the effectiveness of the approach for auditing performance-privacy claims, and highlights the predictive power of the approach relative to experimental results. Finally, we conclude in Section 6.

2 Background and Related Work

This section provides the relevant background on the probabilistic modeling and analysis framework used in this paper, and reviews related work.

To reason about distributional properties of distributed systems we rely on the PMaude [2] probabilistic modeling framework and its

¹<https://github.com/raytheonbnn/maude-hcs>

²The toolchain additionally supports qualitative security and privacy properties expressed in modal and temporal logic by standard model checking.

³The specification of the cover protocol is independent of the tunneling protocol; the latter wraps the former with additional functionality without modifying it.

associated statistical model checking capabilities. PMAude builds on Maude's rewriting logic semantics for concurrent and distributed systems, extending it with time and probability.

Rewriting Logic, Maude and Real-Time Maude. Rewriting logic [40] is a computational logic well suited to specify concurrent systems. A rewrite theory $\mathcal{R}$ specifies: (i) a system's *distributed states* as an *algebraic data type*, and (ii) its *local concurrent transitions* by rewrite rules⁴ $u \rightarrow u'$, where u is a pattern describing certain state fragments, and u' is the pattern describing the resulting local state after the transition. For example, the distributed state of a *generalized actor rewrite theory* [43] $\mathcal{R}$ is a multiset of objects and messages, and its rewrite rules specify its asynchronous communication events such as, for example, a server's response to a client's message, formalized by a rule of the form $(x \triangleleft y : m) \langle x \mid atts \rangle \rightarrow \langle x \mid atts' \rangle (y \triangleleft x : m')$, where $\langle x \mid atts \rangle$ is the current state *atts* of sever x when receiving a message m from client y , which then changes its state to *atts'* and sends back a message m' to y . More generally, the rules in a generalized actor rewrite theory are either: (i) *message-triggered rules* of the form⁵ $(x \triangleleft y : m) \langle x \mid atts \rangle \rightarrow \langle x \mid atts' \rangle \text{ msgs}$, where *msgs* is a (possibly empty) set of messages sent by x , which capture the Actor model [1, 43], or (ii) *object-triggered rules* of the form $\langle x \mid atts \rangle \rightarrow \langle x \mid atts' \rangle \text{ msgs}$, where an actor on its own changes its state and sends messages. For example, the rule's left-hand side $\langle x \mid atts \rangle$ may be a pattern describing the expiration of a timer in x , which may be reset and may cause some messages to be sent. Maude [13] is a high-performance implementation of rewriting logic as a declarative programming language. Concurrent systems with real-time features such as local clocks, timers and timestamps are modeled and analyzed as *real-time rewrite theories* in the Real-Time Maude language extension [49].

Probabilistic Rewrite Theories, PMAude, and Statistical Model Checking. Generalized actor systems are intrinsically nondeterministic, both because of their asynchronous behavior and because of unpredictable network delays. *Probabilistic rewrite theories* [2, 33] make it possible to *quantify* the non-determinism of such systems as follows. Message delays in a given network infrastructure can be modeled by a continuous probability distribution π that agrees with experimental measurements. Then, our above client-server rule becomes the *probabilistic rewrite rule* $(x \triangleleft y : m) \langle x \mid atts \rangle \rightarrow \langle x \mid atts' \rangle \text{ delay}((y \triangleleft x : m'), d)$ with *probability* $d := \pi$. That is, the nondeterministic delay d in the transmission of message $(y \triangleleft x : m')$ becomes now probabilistically quantified by the distribution π from which d is sampled. *delay* $((y \triangleleft x : m'), d)$ describes a message *en route* to y , which "arrives" to y , i.e., it becomes $(y \triangleleft x : m')$, when the system's global clock advances d time units. PMAude [2] extends Real-Time Maude to support *probabilistic (and real-time) generalized actor rewrite theories* of this kind. Since these theories quantify the system's real-time behavior, they make it possible to measure

various system performance properties, such as latency, throughput, and so on. Furthermore, such quantitative properties can be systematically estimated by submitting the P-Maude generalized actor rewrite theory $\mathcal{R}$ to *statistical model checking* (SMC) [3].

However, for SMC to be possible some requirements must be met. First, probabilistic rewrite rules are not directly executable. For example, the choice $d := \pi$ in the above client-server rule is not executable: it must be *simulated* by sampling π . In [33] $\mathcal{R}$ is made executable by the *Sim* transformation $\text{Sim} : \mathcal{R} \mapsto \text{Sim}(\mathcal{R})$, which makes SMC possible on $\mathcal{R}$ by performing *Monte Carlo simulations* of $\text{Sim}(\mathcal{R})$. However, for most SMC tools, $\mathcal{R}$ must be a *purely probabilistic* model. This means that in any Monte-Carlo simulation of $\mathcal{R}$ the probability of reaching a state where two different transitions can be nondeterministically chosen is 0. This is called the *absence of nondeterminism* (AND) property [33].

The AND property problem can be solved by solving a bigger problem, namely, automating the passage from real-time actor rewrite theories to PMAude rewrite theories. For example, automating the passage from a client-server system to its PMAude counterpart. This can be done for any real-time actor rewrite theory $\mathcal{R}$ provided a specification Π assigning a probability distribution π_r to each rule r in $\mathcal{R}$ is given. This is the so-called *P transformation* $P : \mathcal{R} \mapsto \mathcal{R}_\Pi$ in [33], where $\mathcal{R}_\Pi$ is a PMAude actor rewrite theory. As shown in [33], under mild conditions on the input theory $\mathcal{R}$ and on the continuous distributions in Π , $\mathcal{R}_\Pi$ satisfies the AND property and is therefore amenable to SMC analysis. There is, however, a remaining problem: the input theories in [33] are *untimed*, i.e., they do not have any real-time features like clocks, timers, timeouts, etc. This serious limitation⁶ has been recently overcome in [44] by: (i) generalizing P to the *timed P transformation* $P : \mathcal{R} \mapsto \mathcal{R}_\Pi$, where now $\mathcal{R}$ can be a real-time generalized actor rewrite theory, and (ii) proving that, under mild conditions on $\mathcal{R}$ and Π , $\mathcal{R}_\Pi$ satisfies the AND property.

There are still two more issues in the SMC verification of PMAude theories. First, we need a probabilistic temporal logic to express quantitative properties. QuaTex [2] is an expressive *quantitative* probabilistic logic. In standard temporal logic state predicates are Boolean-valued. In QuaTex state properties are *real-valued*, so that when model checking a QuaTex formula φ in a PMAude theory, the result is the estimated value of the property specified by φ , e.g., latency or throughput, etc. Second, it may not be possible to evaluate sophisticated QuaTex properties in the models $\mathcal{R}_\Pi$ or $\text{Sim}(\mathcal{R}_\Pi)$ because the relevant information has not been "logged" in the states. For example, estimating the average delay of all queries in a distributed transaction system requires access to such logs. The user can modify $\mathcal{R}_\Pi$ or $\text{Sim}(\mathcal{R}_\Pi)$ to log relevant events; but this is tedious and error-prone. The *M transformation* in [33], $M : \text{Sim}(\mathcal{R}_\Pi) \mapsto M(\text{Sim}(\mathcal{R}_\Pi))$ automates this last step in a correct-by-construction way. The *P*, *Sim* and *M* transformations automate the entire SMC process of model checking a QuaTex qualitative property on a PMAude theory. QMAude [53] is at present the most efficient SMC tool to verify QuaTex properties on PMAude theories. An example illustrating all the ideas in this section, as well as the

⁴More generally, rewrite rules can be *conditional*, i.e., of the form $u \rightarrow u'$ if *cond*, so that the rule does not fire unless the condition or guard *cond* is satisfied. Henceforth all rules will be silently assumed to be possibly conditional even if *cond* is not mentioned.

⁵For simplicity, actors are here described as objects $\langle o \mid atts \rangle$, where o is the actor's name or identifier, and its state *atts* is a record of attribute-value pairs. But this is an *untyped* view of actors, which can more generally be of the form $\langle o : C \mid atts \rangle$, where C is the actor's *class*, and actor classes are related by class inheritance hierarchies [41].

⁶The limitation is serious because many theories $\mathcal{R}$ in PMAude, including those with real-time features, cannot be expressed as theories of the form $\mathcal{R} = \mathcal{R}_{0\Pi}$ for $\mathcal{R}_0$ an untimed generalized actor rewrite theory.

use of the PMAude pipeline for analyzing the undetectability of HCS systems, is given in Appendix A.

In this work we use the PMAude pipeline to define executable probabilistic semantics for both ordinary and HCS deployments, and to generate samples of observable traces under each model. We leverage statistical model checking to estimate adversary-relevant quantities, such as the true positive rate and false positive rate of a specified detection test, together with confidence intervals. These estimates are then used to derive lower bounds on divergence measures, enabling us to audit claimed undetectability guarantees as described in Section 3.1.

Related Work Recent work has argued that hidden communication systems should be evaluated using formal cryptographic methodologies rather than ad hoc empirical testing. In particular, Pereira et al. [51] propose modeling undetectability as simulation-based indistinguishability between real HCS traces and ideal benign traces and advocate machine-checked proofs of such claims. Our work is complementary. We formalize undetectability as an information-theoretic property of the trace distributions induced by a specified deployment model and develop Maude-HCS as a framework for auditing such claims under concrete system and adversary assumptions. Rather than proving undetectability universally, we instantiate executable probabilistic models and estimate divergence over parameterized observation windows, enabling quantitative evaluation of detectability and its tradeoffs with performance in realistic deployments.

State-of-the-art frameworks for quantitative analysis of distributed systems are typically based on statistical model checking (SMC) [3] or probabilistic model checking (PMC) [5]. Maude-HCS adopts an SMC-based approach over executable rewriting-logic models, trading the exactness of PMC for improved scalability while retaining precise quantitative reasoning through probabilistic temporal logic and Monte Carlo estimation with confidence guarantees. In contrast to automata-based tools such as UPPAAL-SMC [17], PRISM [31], SBIP [38], and the Modest toolset [10, 29], which rely on finite-state models and often require ad hoc encodings to represent distributed protocols with dynamic actors, message queues, and unbounded data structures, Maude-HCS operates directly on expressive actor-style rewrite theories. This enables faithful modeling of concurrent networked systems and supports both qualitative and quantitative analysis within a single semantically aligned executable specification, avoiding the modeling inconsistencies that can arise when separate abstractions are used for different forms of analysis.

Recent work by Nelson et al. [48] also studies formal guarantees for metadata privacy in real-world messaging protocols. They model deniable messaging over Signal using information-flow techniques and prove qualitative noninterference properties for a specific protocol design. Our work is complementary. Maude-HCS provides a general framework for quantitatively auditing the statistical indistinguishability of modeled HCS deployments using divergence bounds.

3 Formalizing Undetectability of Hidden Communication Systems

At a high level, an HCS seeks to embed messages into ordinary network activity in such a way that an external observer cannot

reliably determine whether hidden communication is taking place. Conceptually, there are two worlds of execution

- an *ordinary* world Ord in which the baseline protocol executes without hidden communication
- a *hidden communication* world HCS in which the same protocol carries covert messages

Undetectability is defined in terms of the interactions between an observer and each of these worlds.

Adversary model. We let $\mathcal{O}$ denote the alphabet of protocol observables such as packet sizes, timings, and directions. We let Act denote the set of actions available to the adversary such as dropping packets, delaying messages, or influencing protocol behavior. We let $\mathcal{E}$ denote the set of environment states such as network queues, CPU load, and background traffic conditions.

An adversary $\mathcal{A}$ is an interactive possibly randomized algorithm that, given a transcript prefix over the protocol’s execution, outputs the next action in Act .

Interactive execution. Execution is modeled as an interactive protocol between a world $X \in \{\text{Ord}, \text{HCS}\}$, an environment state $E \in \mathcal{E}$, and an adversary $\mathcal{A}$. At time $\tau_0 = 0$, the environment is initialized to some state $E_0 \in \mathcal{E}$.

The execution proceeds as a sequence of events at continuous times $0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_k \leq t$. At each event time τ_i , the adversary selects an action $A_i \in \text{Act}$, and the world evolves according to its transition semantics, producing an observable $O_i \in \mathcal{O}$ together with an updated environment state E_i .

A transcript up to time horizon t containing k events is the sequence

$$\pi_t = ((E_0, \tau_0, A_0, O_0), (E_1, \tau_1, A_1, O_1), \dots, (E_k, \tau_k, A_k, O_k))$$

Adversary view and induced trace distributions. Fix a world $X \in \{\text{Ord}, \text{HCS}\}$, an adversary $\mathcal{A}$, and an initial state $E_0 \in \mathcal{E}$. For each such choice, the interaction between X and $\mathcal{A}$ over horizon t defines a probabilistic experiment where we let $\text{View}_{\mathcal{A}, E_0, t}^X$ denote the random variable corresponding to the adversary’s *view* of the interaction, defined as the projection of the full transcript onto the observable interaction:

$$\text{View}_{\mathcal{A}, E_0, t}^X = ((\tau_i, A_i, O_i))_{i=0}^k$$

Undetectability. Hidden communication is undetectable if no adversary interacting with either world can reliably distinguish which world generated its observed transcript.

Definition 3.1 (($\mathcal{M}, d$)-Undetectability). Fix an observation horizon t . Let $\mathcal{M}$ be a divergence or distance measure on pairs of probability distributions over transcripts of length t , and let d be a parameter in the range of $\mathcal{M}$. A deployment satisfies ($\mathcal{M}, d$)-HCS undetectability up to horizon t if $\forall \mathcal{A}, \forall E_0 \in \mathcal{E}$,

$$\mathcal{M}(\text{View}_{\mathcal{A}, E_0, t}^{\text{HCS}}, \text{View}_{\mathcal{A}, E_0, t}^{\text{Ord}}) \leq d$$

This definition admits multiple operational interpretations depending on the choice of $\mathcal{M}$.⁷ For example, when $\mathcal{M}$ is KL divergence $D_{\text{KL}}(P||Q) = \mathbb{E}_{x \sim P} \left[\log \frac{P(x)}{Q(x)} \right]$, $(\mathcal{M}, d)$ -Undetectability bounds the adversary's expected log-likelihood ratio, that is, its expected information gain from observing a length- t trace. For divergence measures that admit hypothesis-testing interpretations, classical inequalities relate divergence to testing performance. In particular, when $D_{\text{KL}}(\text{View}_{\mathcal{A}, E_0, t}^{\text{HCS}} || \text{View}_{\mathcal{A}, E_0, t}^{\text{Ord}})$ is small, no detector can simultaneously achieve a low false positive rate and a substantially higher true positive rate. Other probability distance measures, such as total variation distance or max divergence, admit analogous operational guarantees.

3.1 Auditing Undetectability via Monte Carlo Trace Sampling

Definition 3.1 characterizes undetectability in terms of a divergence $\mathcal{M}(Q_t, P_t) = \mathcal{M}(\text{View}_{\mathcal{A}, E_0, t}^{\text{HCS}}, \text{View}_{\mathcal{A}, E_0, t}^{\text{Ord}})$ between the trace distributions induced by the ordinary and HCS worlds, quantified over all adversary strategies and initial environmental conditions. We show how Maude-HCS can be used to *audit* claims of undetectability by providing a high-probability lower bound on $\mathcal{M}(Q_t, P_t)$ from Monte Carlo samples. Such an audit is necessarily carried out for a specific instantiated model, adversary, execution horizon t , and distribution over initial environmental conditions, as determined by the executable semantics of the system under consideration. In this work we focus on KL divergence as the undetectability measure, though the same methodology applies to other divergence notions such as max divergence and total variation distance. A certified lower bound on $D_{\text{KL}}(Q_t || P_t)$ can be used directly to audit undetectability claims: if a deployment claims (D_{KL}, d) -HCS undetectability, meaning that $D_{\text{KL}}(Q_t || P_t) \leq d$, then any certified lower bound exceeding d falsifies the claim under the specified modeling assumptions, whereas a lower bound below d does not refute it.

We emphasize that this methodology provides only a lower bound on $D_{\text{KL}}(Q_t || P_t)$. Passing such an audit does not establish an upper bound on the true divergence, nor does it prove that the claimed privacy parameter holds universally. Rather, it provides evidence consistent with the claim under the sampled executions, adversary model, and environmental assumptions encoded in the executable semantics.

Auditing Framework. Although undetectability is defined in terms of divergence between trace distributions, directly estimating these distributions is infeasible in many practical settings because observable traces are inherently high dimensional. For example, a passive adversary may observe the full sequence of inter packet arrival times over a window of length t , producing a vector whose dimension depends on the number of packets transmitted. Directly estimating the full distributions P_t and Q_t over such spaces is statistically infeasible, since discretizing a d -dimensional observable with B bins per coordinate yields a joint domain of size B^d . As

⁷We do not require $\mathcal{M}$ to be symmetric. Many divergences with meaningful operational interpretations are inherently asymmetric. The ordering $\mathcal{M}(\text{View}_{\mathcal{A}, E_0, t}^{\text{HCS}}, \text{View}_{\mathcal{A}, E_0, t}^{\text{Ord}})$ reflects the detection-oriented setting in which the adversary seeks evidence that an observed transcript was generated by the HCS world rather than the ordinary baseline.

a result, the number of Monte Carlo samples required to reliably estimate divergence becomes prohibitively large.

In this regime we instead audit detectability through an explicit adversarial test. Let $\mathcal{O}_t$ denote the space of all valid observable traces up to time horizon t . Let $\varphi: \mathcal{O}_t \rightarrow \{0, 1\}$ be any detector that attempts to distinguish HCS traces from ordinary traces. Its false positive and true positive rates are

$$\text{FPR}_\varphi = \mathbb{E}_{o \sim P_t} [\varphi(o)]$$

$$\text{TPR}_\varphi = \mathbb{E}_{o \sim Q_t} [\varphi(o)]$$

Let $Y = \varphi(O_t)$ denote the detector's one bit output, where $O_t \in \mathcal{O}_t$ is the observed trace. Under P_t , the random variable Y is distributed as a Bernoulli random variable $\text{Bern}(\text{FPR}_\varphi)$, and under Q_t it is distributed as $\text{Bern}(\text{TPR}_\varphi)$. By the data processing inequality for KL divergence,

$$D_{\text{KL}}(Q_t || P_t) \geq D_{\text{KL}}(\text{Bern}(\text{TPR}_\varphi) || \text{Bern}(\text{FPR}_\varphi))$$

Equivalently,

$$D_{\text{KL}}(Q_t || P_t) \geq \text{TPR}_\varphi \log \frac{\text{TPR}_\varphi}{\text{FPR}_\varphi} + (1 - \text{TPR}_\varphi) \log \frac{1 - \text{TPR}_\varphi}{1 - \text{FPR}_\varphi}$$

Estimating TPR_φ and FPR_φ from n Monte Carlo samples in each world yields confidence intervals

$$\text{TPR}_\varphi \in [T_L, T_U], \quad \text{FPR}_\varphi \in [F_L, F_U]$$

that hold jointly with probability at least $1 - \delta$. A conservative high-probability lower bound is obtained by minimizing the Bernoulli KL divergence over all parameters consistent with these intervals. Since $D_{\text{KL}}(\text{Bern}(q) || \text{Bern}(p)) \geq 0$ with equality iff $p = q$, this minimum is attained by choosing p and q as close as possible. In particular, with probability at least $1 - \delta$, we have:

$$\begin{aligned} D_{\text{KL}}(Q_t || P_t) &\geq \min_{q \in [T_L, T_U], p \in [F_L, F_U]} D_{\text{KL}}(\text{Bern}(q) || \text{Bern}(p)) \\ &= \begin{cases} 0 & \text{if } [T_L, T_U] \cap [F_L, F_U] \neq \emptyset, \\ D_{\text{KL}}(\text{Bern}(T_L) || \text{Bern}(F_U)) & \text{if } T_L > F_U, \\ D_{\text{KL}}(\text{Bern}(T_U) || \text{Bern}(F_L)) & \text{if } T_U < F_L \end{cases} \end{aligned}$$

We report this minimized value as our certified lower bound.

Operational interpretation. The KL divergence $D_{\text{KL}}(Q_t || P_t)$ equals the adversary's expected log-likelihood ratio in favor of the HCS world after observing a length- t trace. If $D_{\text{KL}}(Q_t || P_t) = \varepsilon$, the expected log-odds increase by ε . For example, if the adversary assigns prior probability 0.01 to HCS activity, then the prior odds are $0.01/0.99 \approx 0.0101$. If $\varepsilon = 1$, the expected posterior odds become $0.0101 \cdot e \approx 0.0275$, corresponding to posterior probability approximately 0.027. Thus, KL divergence directly quantifies the expected multiplicative shift in posterior odds induced by a single length- t observation window.

4 Modeling and Analysis Workflow

Maude-HCS enables designers to model their HCS design, and to evaluate performance and privacy guarantees about their model. Designers can ask, for example, *what is the goodput of a hidden communication system design for a desired probability of detection?*

The modeling and analysis workflow is shown in Figure 1. The model of the underlying network and protocols without hidden

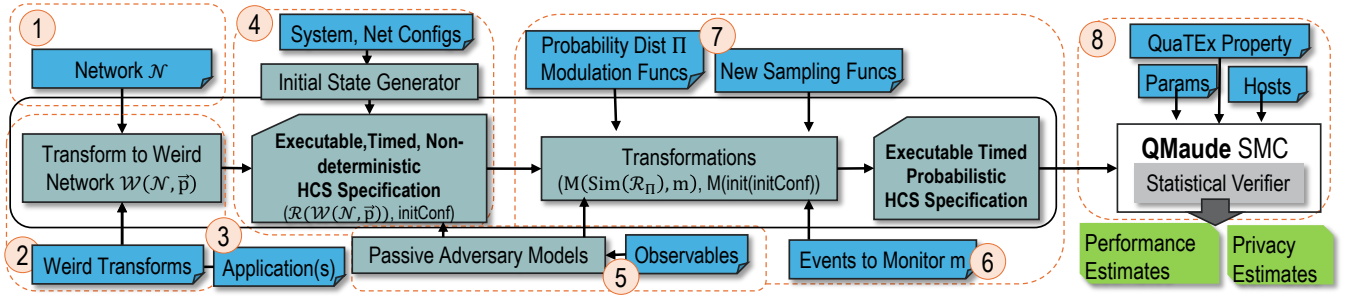


Figure 1: Maude-HCS reduces the burden on the designer while ensuring semantic consistency. Designers build nondeterministic models of the underlying network ①, of the weird transforms ②, and of the application ③ as generalized actor theories. They also provide the actor and system configs ④ which are used to generate the initial HCS configuration. Designers then specify the adversary observables ⑤ (and extend the adversary detection logic as needed), and the performance relevant events to monitor ⑥. Maude-HCS generates executable, timed, and probabilistic specification used for statistical analysis. For analysis, designers specify the quantitative performance and privacy properties of interest and the SMC parameters α, δ (and hosts for distributed SMC) ⑧. These are used to produce performance and privacy guarantees.

communication, $\mathcal{N}$, is directly formalized in Maude as a generalized actor rewrite theory that can be analyzed as an independent entity. A formal pattern [42] transformation $\mathcal{N} \mapsto \mathcal{W}(\mathcal{N}, \tilde{p})$ takes the underlying network system $\mathcal{N}$ and transforms it to a new weird network system.⁸ The parameters $\tilde{p}$ typically include a set of one or more transformations that embed/extract hidden information. The network system $\mathcal{N}$ is preserved as a subsystem of the weird network $\mathcal{W}(\mathcal{N}, \tilde{p})$. The $\mathcal{N} \mapsto \mathcal{W}(\mathcal{N}, \tilde{p})$ transformation wraps each selected actor object in $\mathcal{N}$ inside a corresponding meta-actor, which hides its communications with other objects in $\mathcal{N}$ *without changing the original object's behaviors*. Using the M transformation [33], a monitor object records selected events during execution that are relevant for computing by SMC performance measures specified as QuaTex form. Similarly, we implement an adversary object that records selected observables during execution that are relevant for detectability analysis based on a model of the adversary's capabilities. Monitored events and adversary observables are programmed using patterns (e.g., filtering the messages the passive adversary can observe). Appendix A illustrates this modeling approach using a simple HCS example.

Modeling boundary. The designer specifies the operational model, including the baseline protocol, HCS transformation, initial configurations, probabilistic assumptions, monitored events, and adversary observables. This step is not currently automated. Once this boundary is fixed, Maude-HCS automates the remaining analysis pipeline by generating executable timed probabilistic specifications and using QMaude/QuaTEx to estimate the performance and detector quantities used in the audit.

As discussed in Section 2, an HCS model begins with a non-deterministic timed rewrite theory $\mathcal{R}$, which is transformed into a probabilistic timed rewrite theory $M(\text{Sim}(\mathcal{R}_\Pi))$, in which sources of uncertainty, such as message delays, are modeled using explicit probability distributions, and distributions are simulated via sampling. As a result, $\text{Sim}(\mathcal{R}_\Pi)$ defines a well-defined probability measure over execution traces. There is no residual nondeterminism,

⁸We refer to the resulting hidden communication system as a weird network.

and each execution is fully determined by the sampled random values.

Ordinary and HCS worlds. Our modular modeling approach allows us to easily distinguish between two executable systems. The ordinary system is represented by $M(\text{Sim}(\mathcal{R}(\mathcal{N})_\Pi))$, and does not contain the weird transforms or the HCS application actors (this is the system without hiding). On the other hand, the HCS system is represented by $M(\text{Sim}(\mathcal{R}(\mathcal{W}(\mathcal{N}, \tilde{p}))_\Pi))$ and includes the weird network behaviors and the HCS application. These two rewrite theories share the same baseline protocol structure and differ only in whether hidden communication is embedded. Both systems are instrumented with identical monitors, so that they expose observables in a common domain $\mathcal{O}$.

Executing each system samples a probability distribution over $\mathcal{O}_t$. We denote these distributions by P_t for the ordinary system and Q_t for the HCS system. These distributions are induced by the probabilistic semantics of the corresponding rewrite theories, including any randomized initialization of system and environment state specified by the model.

Undetectability. At a high level, Maude-HCS serves two purposes. First, it defines a precise notion of what a passive adversary can observe with its observational capabilities by explicitly instrumenting executions to record observable traces. Second, it enables sampling from the induced distributions over those traces, which can then be compared using statistical techniques. Much of the underlying infrastructure already exists in PMaude and QMaude. Our contribution in this paper is in developing the theoretical foundations of undetectability analysis for HCS systems, and in extending the Maude-HCS infrastructure to support privacy analysis, where the object of interest is inherently relational and distributional. Rather than checking temporal properties of a single system execution, we are interested in comparing the *distributional behavior* of two related systems: an ordinary deployment and an HCS deployment.

Maude-HCS provides a workflow for instrumenting executable probabilistic models so that such comparisons can be carried out in a principled and repeatable manner.

Once instrumented, each model is analyzed using statistical model checking. Rather than exporting raw traces and performing offline estimation, we specify adversary-relevant quantities—such as the probability that a given detector alarms within horizon t —as quantitative QuaTex properties. QMaude then estimates these probabilities via Monte Carlo simulation with user-specified confidence parameters. In particular, for a fixed detector φ , we directly obtain high-confidence estimates of TPR_φ and FPR_φ under the HCS and ordinary models, respectively. These SMC-derived estimates are then used to compute the certified lower bound on $D_{\text{KL}}(Q_t \| P_t)$ described in Section 3.1. In this way, divergence-based undetectability claims are audited through statistically verified detector behavior within each executable model.

Performance In the context of a HCS, quantitative performance measures such as latency, goodput, and throughput can be precisely defined as QuaTex formulas. For example, if HCS user Alice uses the HCS to transfer a file of size B bytes to user Bob and it takes T seconds, the goodput is B/T bytes/second. These quantitative measures can be precisely specified using QuaTex in Maude-HCS. Maude-HCS SMC provides a statistical guarantee of the form *the expected value of the transfer time is within a range $[\hat{v} - \delta/2, \hat{v} + \delta/2]$ with confidence $(1 - \alpha)$* , given desired confidence parameters α and δ [2]. Throughout the evaluation, we always set $\alpha = 0.05$ (95% confidence). The designer can thus characterize the performance of the design, by assessing the effect of the different design parameters on performance. For example, *how does link loss (or MTU size) affect file transfer time?* This type of performance characterization can yield insights for improving the design, and informing the deployment.

5 Evaluation

We conduct a thorough evaluation of our approach on a two-channel HCS that combines a DNS-tunneled control channel, a Mastodon-based steganographic data channel, background DNS-/Mastodon traffic generators, a NAT/router boundary, lossy links, and post-NAT Zeek-based passive monitoring. This case study is substantially more complex than a toy protocol while still small enough to support controlled parameter sweeps. We quantify undetectability and performance across a range of scenarios and adversary capabilities, and assess predictive power by validating model guarantees against a testbed implementation of the same system.

We structure our evaluation around three sets of experiments.

Privacy-Performance Trade-off Analysis: Given an adversary detection capability, we conduct a privacy-performance trade-off analysis across a range of scenarios. To do so, we increase application performance and audit KL divergence between ordinary and HCS distributions across a range of observables.

Sensitivity Analysis: We vary the (1) initial configuration by instantiating different scenarios (e.g., by varying link loss, and background traffic), (2) adversary classifier thresholds, and (3) application parameters (e.g., number of files Alice is sending). We

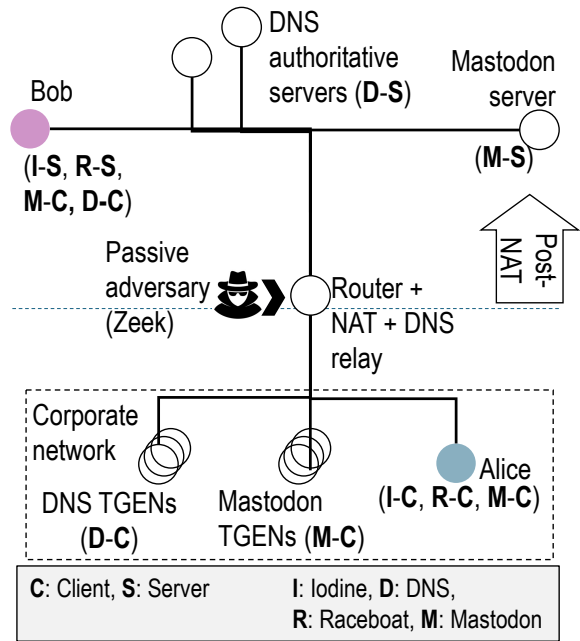


Figure 2: The HCS system and network modeled and instantiated on the testbed, and used for evaluation.

observe the effect on both performance and undetectability.

Semantic Alignment: We demonstrate how our model-based results align closely with experimental results from the testbed implementation, attesting to the predictive power of the Maude-HCS models and guarantees.

5.1 HCS System and Experimental Setup

We model and instantiate a tunneling-based two-channel HCS, comprising of a control channel, and a data channel. The control channel is realized using Iodine [24] tunneling over the Domain Name System (DNS) protocol [45]. The data channel is realized using Destini [15, 57] tunneling over the Mastodon social network [35], the latter serving as a whiteboard. Figure 2 illustrates the system and the network. HCS user Alice, located within the corporate network, covertly sends files to Bob located outside the corporate network.

5.1.1 Testbed Experimental Setup. The control channel is operated as a QUIC connection over Iodine [24], a DNS tunneling protocol used to bypass firewalls by encapsulating traffic into DNS queries. We fitted Alice with an Iodine client, and Bob with an Iodine server. The data channel of the HCS is operated as Destini-over-Mastodon: covert traffic is segmented into byte chunks, which Destini, a steganographic protocol, encodes into cover images. These images are then handled by Raceboat [58] to upload onto a Mastodon server as media attachments in *toots*, which contain hashtags that the server side of the HCS can use to retrieve and then decode covert data. Raceboat acts as a Pluggable Transport that handles encoded and benign image uploads and downloads.

The HCS operates over an underlying network that would look familiar to network operators: Alice and synthetic traffic generation sources (TGENs) operate on a corporate network behind a router that translates network addresses (see dotted box in Figure 2). This is colloquially referred to as NAT, and pre- and post-NAT activity applies to corporate versus all other traffic. Finally, DNS infrastructure is further emulated with public and domain resolvers.

DNS and Mastodon traffic generators mimic real deployments where multiple users utilize common resources, like a Mastodon server or DNS resolvers, alongside the HCS. Background synthetic traffic ends up mixing with HCS traffic and complicates detection.

Passive Adversary The (passive) adversary is located post-NAT as a monitor and can observe all traffic ingressing and egressing the corporate network (from traffic generators and Alice). The adversary is instantiated on the testbed using the Zeek monitoring tool [50, 56], with a range of detectors. The detectors were instantiated with thresholds based on baseline data from which covert traffic is absent. For each detector, when the threshold is exceeded an alarm is recorded in Zeek. The adversary has three detectors monitoring post-NAT corporate traffic, two are cumulative (C) and one is a moving average (MA).

- **DNS query count ("C2"):** the total number of observed DNS queries is compared to a baseline threshold.
- **HTTPS request count ("C8"):** the total number of observed HTTPS requests is compared to a baseline threshold.
- **DNS query rate ("MA1"):** the moving average DNS query rate is compared to a baseline threshold. The query rate is computed over a 60s moving window, starting with baseline queries; when the average rate exceeds $k * R$, where k is a multiplier and R is a well-chosen threshold, for n consecutive bins, the alarm triggers.

5.1.2 Note on Testbed Implementation. The testbed matching Figure 2 was developed by an independent test and evaluation team and deployed as a Dockerized network. The testbed recreates dynamic network conditions using Linux Traffic Control to emulate link delays and per-packet loss. The deployed software stack also contains implementation artifacts that are not part of the protocol-level model. For example, Raceboat occasionally creates spurious TCP connections, and both Raceboat and Iodine incur computation and scheduling delays that depend on the containerized runtime. We use “bug” in this implementation sense: the testbed does not always exhibit the clean protocol behavior that the model abstracts, and some discrepancies between Maude-HCS and the testbed are best explained by these implementation-level events rather than by a mismatch in the modeled HCS design.

5.1.3 Corresponding Actor Models. We developed a formal faithful model of the system, including all the actors for both channels and for the application, and for the underlying network, as well as their configurations. The Iodine-DNS channel uses DNS as the underlying network; we started with the formal DNS model of [32], and added the Iodine weird network including the Iodine client and server supporting bi-directional communications, as well as the DNS traffic generators. The Destini-Mastodon channel uses the Mastodon infrastructure as the underlying network where we

implement mastodon client and server functionality for posting and fetching toots and images, as well as mastodon background traffic generators. The weird network was implemented using the raceboat client and server with Destini module covertly encoding application data into images uploaded to mastodon.

For the HCS application, we implemented Alice as the sending application which encapsulates an iodine client, and a raceboat client, and mastodon client used by raceboat. Bob is the receiving application, which encapsulates the iodine server and the raceboat server, and a mastodon client used by raceboat. In terms of infrastructure, we implemented a router that separates the corporate from the public network, and the initial state generator that takes experimental configurations and converts them into an initial configuration for analysis. Several of the actors actions are driven by *markov processes*, including the DNS and mastodon traffic generators, raceboat client and server mastodon clients. The markov processes have two simple states for posting and fetching media toots. The post or fetch action leads to a wait state during which the underlying Mastodon client undertakes all API-level actions to upload or download media images (encoded or not) and status toots. Waits in the wait state are normally distributed with a mean and standard deviation specified by the configuration.

Adversary and Undetectability. We implemented the passive adversary actor including the observables (as patterns over messages). We implement the three detectors, “C2”, “C8” and “MA1”, as mixed properties in QuaTEX over the adversary configuration (which captures the trace observables), and quantify their TPR and FPR of each detector by measuring probability of alarm relative to the HCS (TPR) or to the ordinary system (FPR). In practice, the adversary’s detector capability can be abstracted as a binary classifier that takes high dimensional trace observables as input and produces a binary detection decision (and in the case of detection, additional metadata such as the time $t' \leq t$ at which detection occurs). Any such classifier has a TPR and FPR from which we compute a lower bound on KL divergence (Section 3.1). While in this evaluation we experiment with the three detectors, any such classifier can be audited.

5.1.4 Quantitative Properties. We measure the following HCS application performance properties, expressed as QuaTex properties.

- **Latency:** The delay to deliver the covert files from the moment Alice starts sending the first one, to the moment Alice receives receipt acknowledgment from Bob that the last file was delivered. Note that Alice will not begin to send the next file until it receives an acknowledgment from Bob on the control channel that the previous file was delivered.
- **Goodput:** The total file bytes successfully delivered divided by the latency.

The Monitor object in our models keeps track of relevant events required for computing the performance properties. We additionally measure the following privacy-related properties, from the perspective of the adversary.

- **Operation (Op) duration:** The total experiment time elapsed before the adversary detector alarms. Each detector (C2, C8, MA1) may produce a different op duration value. The op

Scenario #	Loss Alice	Loss Bob	Num files	Total Bytes	Num backgrnd generators
*1	0.0%	0.0%	10	6,820B	16
2	0.0%	0.0%	10	57,509B	32
3	0.0%	0.0%	10	57,123B	48
4	5.0%	5.0%	5	8,622B	16
5	5.0%	5.0%	5	20,758B	32
6	5.0%	5.0%	5	27,486B	48
*7	2.5%	2.5%	5	7,463B	16
8	2.5%	2.5%	7	34,301B	32
9	2.5%	2.5%	7	39,030B	48

Table 1: Summary of the test scenarios used for the evaluation audits. Loss Alice is the loss on the link between the router and the Mastodon server / public DNS resolver. Similarly, Loss Bob is loss on the link between Bob and the Mastodon server / public DNS resolver.

duration is less or equal than than latency when there is a detection (and discarded otherwise). Op duration is naturally expressed in QuaTEX as a *mixed property* (example in Appendix B).

- **True Positive Rate (TPR) and False Positive Rate (FPR):** For any input configuration including the adversary detector and its parameters, and resulting trace of length t , we measure the probability of alarm on both the ordinary and the HCS systems. These become the FPR and TPR measures, respectively.⁹ The TPR and FPR estimates are used to compute the lower bound on KL divergence (section 3).

5.1.5 Scenarios. We use the scenarios to exercise the same HCS under three deployment regimes with increasing background activity. All scenarios test covert transmissions of files from Alice to Bob. Scenarios 1–3 have no packet loss and isolate the detectability cost of covert traffic when modest delay is the main network stress. Scenarios 4–6 add 5% packet loss on the Alice-side and Bob-side links to the public DNS resolver and Mastodon server. Scenarios 7–9 use the same setup with 2.5% packet loss. Within each group, the number of DNS and Mastodon traffic generators increases from 16 to 32 to 48, so the scenarios measure how additional cover traffic affects false positives, true positives, latency, and goodput. Table 1 summarizes the parameters.

5.2 Privacy-Performance Trade-off Analysis

We quantify undetectability–performance tradeoffs across three representative scenarios (Scenarios 1, 4, and 7), which differ in network loss and background traffic intensity (Table 1). In each scenario, Alice sends a fixed set of covert files while we vary a single application parameter, the mean client wait time between successive HTTP requests in Alice’s Markov posting model. Decreasing this wait time increases the rate at which covert traffic is injected and therefore increases goodput.

⁹For any sampled trace, if the alarm fires, we measure a 1 and a 0 otherwise.

For each configuration, we audit detectability using a fixed passive detector φ and estimate its true positive rate and false positive rate under the HCS and ordinary models, respectively. In this experiment, φ is the cumulative C8 detector, which alarms when the number of observed HTTPS requests exceeds a fixed threshold N within the observation horizon (Section 5.1.1). We convert the resulting SMC estimates of TPR_φ and FPR_φ into a certified lower bound on $\text{KL}(Q_t \| P_t)$ using the auditing procedure in Section 3.1. Figure 3 plots, for each scenario, the certified KL lower bound as a function of the achieved goodput.

Across all three scenarios, increasing goodput generally increases statistical distinguishability between HCS and ordinary executions. Intuitively, more aggressive sending shifts the detector-relevant traffic statistics away from their ordinary distribution, which increases TPR_φ relative to FPR_φ and therefore increases the certified KL lower bound. The scale of the bound varies substantially across scenarios, reflecting the dependence of detectability on network conditions and background traffic.

These results also illustrate how divergence-based undetectability claims can be audited by sweeping a deployment parameter. Suppose a deployment claims to provide $(\text{KL}, 0.01)$ -undetectability under a specified scenario configuration (e.g., Scenario 7) at a given operating point. If the certified lower bound in Figure 3 exceeds 0.01 at that operating point, the claim is falsified under the modeling assumptions and detector specified in Sections 5.1.1 and 3.1. Conversely, a lower bound below 0.01 does not prove undetectability; it only shows that this detector did not produce enough evidence to refute the claimed bound in that regime.

Finally, we observe that the KL divergence lower bound is very high for the other scenarios for the C8 detector, and we do not show these results. This is expected given that the HCS is not private (and adversary can easily distinguish between ordinary and HCS even with these simple detectors).

Detector parameter sweep. In Figure 4, we sweep the moving-average detector threshold multiplier k , which corresponds to evaluating a family of statistical tests with different operating characteristics. Each choice of k induces a different binary classifier φ_k with its own TPR_{φ_k} and FPR_{φ_k} , and therefore its own lower bound on $\text{KL}(Q_t \| P_t)$. The key observation is that different thresholds produce substantially different KL lower bounds. In particular, more conservative thresholds (larger k) reduce the false positive rate and can yield significantly larger KL lower bounds, even if the true positive rate also changes. Since each threshold defines a valid detector, we may view this sweep as auditing the HCS against a suite of statistical tests. An audit can then report the maximum lower bound across this family as the strongest divergence evidence obtained from the tested detectors.

Application parameter sweep. In Figure 5, we vary Alice’s performance by changing the mean wait time (sec) between successive posts in Alice’s Markov model. Longer waits between posts increase the time required to send a file and therefore reduce goodput. In this experiment, the cumulative C8 detector threshold is fixed. For each wait time, we measure the certified KL divergence lower bound (left), the probability of alarm for the C8 detector (middle), and the operating duration until detection (right).

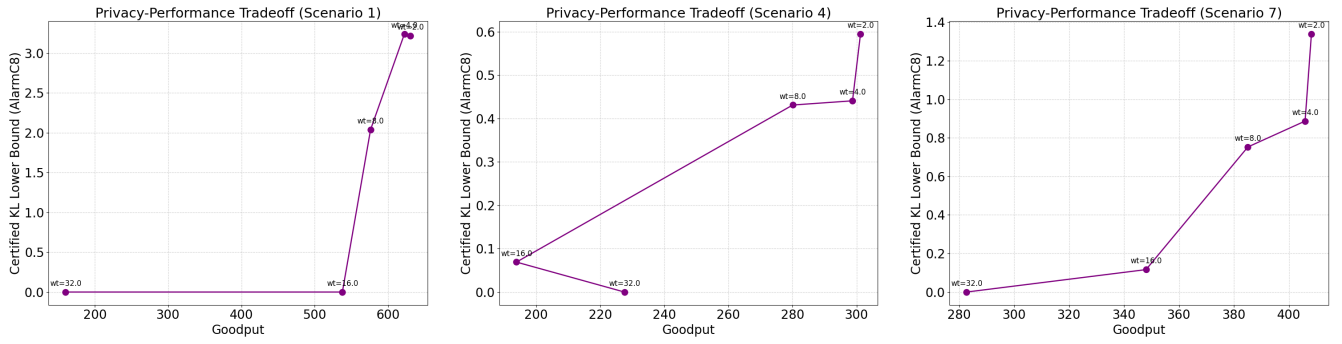


Figure 3: Privacy-performance tradeoff across Scenarios 1, 4, and 7. Each subplot corresponds to a different network condition (see Table 1). For each scenario, Alice’s mean wait time between posts in the Markov model is varied while all other parameters are fixed. Each point shows the achieved goodput (bytes/sec) and the corresponding certified lower bound on $KL(\text{View}^{\text{HCS}}|\text{View}^{\text{Ord}})$, derived from SMC estimates of detector TPR and FPR. Across scenarios, decreasing the wait time generally increases goodput and generally increases statistical distinguishability between HCS and ordinary executions, though the magnitude of the KL bound depends on network loss and background traffic.

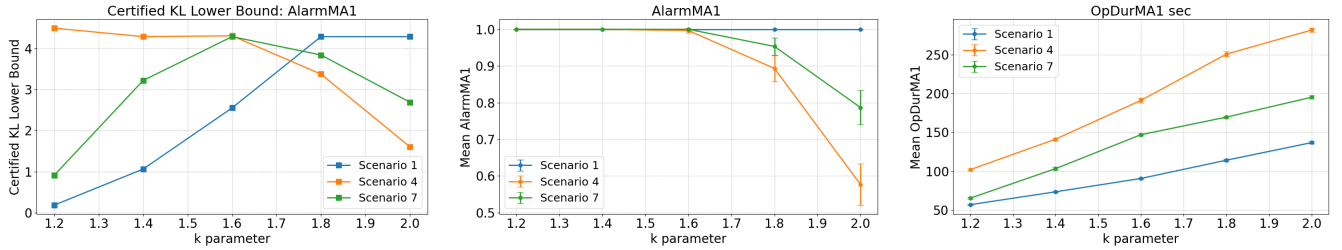


Figure 4: We vary the moving average detector threshold multiplier k (section 5.1.4) and observe the effect on KL divergence bounds (left), on probability of alarm for the moving average detector MA1 (middle), and on operating duration (right). The bar around the alarm datapoints is the radius.

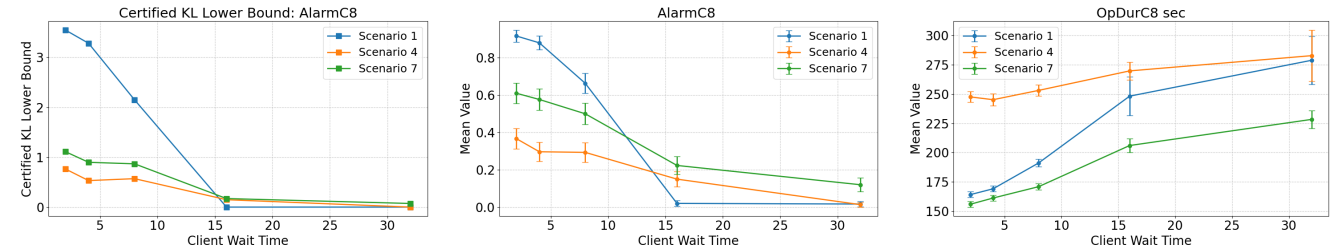


Figure 5: We vary the mean wait time (sec) in Alice’s markov model; longer waits between posts means lower performance. We observe the effect on KL divergence bounds (left), on probability of alarm for the cumulative C8 detector (middle), and on operating duration (right). The bar around the datapoints is the radius. In the KL bounds figure on the left, scenario 1 and 4 datapoints overlap. The threshold $N = 80$ for the C.8 detector is fixed.

As Alice becomes more aggressive, corresponding to shorter wait times, the detector achieves greater separation between TPR and FPR, which produces larger certified KL lower bounds. Conversely, as the wait time increases and traffic more closely resembles ordinary behavior, the KL lower bound decreases. The operating

duration increases with wait time, reflecting that less aggressive sending delays or reduces detection events.

The key takeaway is that divergence is determined jointly by application behavior and adversarial test configuration. Even with a fixed detector threshold, performance tuning alone can move the

system between operating regimes that yield large certified distinguishability and regimes that yield substantially smaller divergence evidence.

5.3 Semantic Alignment Validation

The core of this audit consists in comparing metrics collected from real systems on a testbed against those derived from SMC analysis of our model.

5.3.1 Quantification. Our first effort to quantify semantic alignment focuses on the Iodine control channel only: we modeled and derived expected values of latency and goodput metrics for elemental Iodine behavior. Our goal is to validate Iodine performance guarantees, *i.e.*, to demonstrate empirically that model guarantees transfer to Iodine as actually implemented. We begin by characterizing the latency to transfer a 1,600B file. Given maximum query sizes of 100B, the file will require 16 DNS (encoded) queries. Figure 6 (left) shows close agreement between our model and empirical measurements, with around 26ms difference in the means, otherwise similar distributions, and very low KL divergence (near zero).¹⁰ With tight alignment from the performance metrics we measure like latency and goodput, we can observe that exfiltration latency is driven by the number of queries required to send the file in the absence of loss on the links, and loss and retransmissions in the presence of loss. Figure 6 (right) shows the normalized KL divergence relative to file size. Relative KL divergence decreases and remains very low with increasing file size.

We further substantiate semantic alignment of the entire HCS (Iodine and Destini-Mastodon) by comparing metrics extracted from the testbed and Maude’s HCS model SMC results across the scenarios of Section 5.1.5. For each scenario, empirical results are aggregated from 1,024 runs transmitting between five and ten files. SMC results are the expected value of the metrics after 300 Monte Carlo simulations. These sample counts reflect a deliberate cost–precision tradeoff. Increasing the number of Monte Carlo simulations would shrink the confidence radius, but would also scale analysis time roughly linearly with the number of complete executions. We therefore use 300 simulations for the whole-system scenario sweeps to keep parameter exploration tractable, while using larger samples in the smaller Iodine-only study where individual runs are cheaper.

Figure 7 shows that the Maude model tracks the testbed metrics well, with our model mostly falling within one standard deviation of the testbed-collected statistics for latency and goodput, and at most 13.6% apart for latency, and 17.1% for goodput. For the Operating duration (time-to-detect) metric (Figure 7, bottom left), our model aligns with empirical data well. An audit of the gap revealed that the unmodeled factors like compute delays and spurious TCP connections from Raceboat account for a deviation at least equal to the gap between SMC and empirical results. Figure 7 bottom right shows the same alignment for all scenarios.

6 Discussion and Conclusions

In this paper we introduced the Maude–HCS executable modeling and analysis framework and described its use for analyzing performance and indistinguishability properties of hidden communication

¹⁰This KL divergence is empirically computed on two distributions each characterized by a mean and a standard deviation, and is only intended as a proxy for distance.

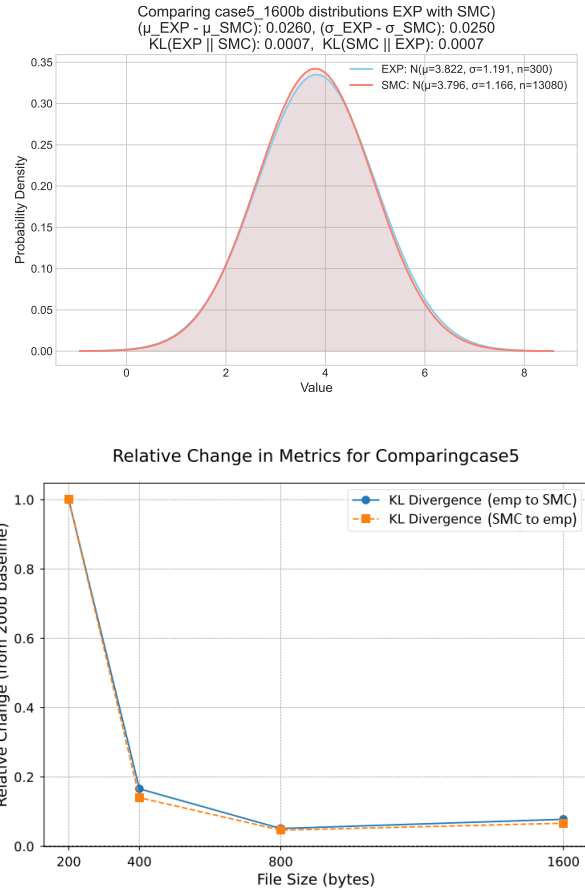


Figure 6: Close Iodine alignment from the latency distributions between Maude model of Iodine and empirical measurements (top), the normalized KL divergence as the file to transmit increases in size 1,600B (bottom).

systems. We propose a method to quantify indistinguishability of hidden communications from the underlying communications. We use this to develop a formal method to audit measures of the differences between the observations of the hidden communications for a protocol $W[\mathcal{R}]$ and the analogous observations for the underlying protocol $\mathcal{R}$ which is used by $W[\mathcal{R}]$ to hide its communications.

We evaluate the Maude–HCS framework, auditing and tradeoff methods on a collection scenarios representing real-world hidden communication systems, including DNS tunneling and posting pictures with embedded information on a whiteboard such as Mastodon.

The models presented here were developed in a project where the aim was to demonstrate that formal models of existing hidden communication system implementations can provide realistic performance and detectability analyses, thus reducing the need for lengthy testbed simulations. Beyond the basic challenge of modeling such complex systems, this meant capturing the behavior of existing implementations that are full of quirks, unspecified features, and bugs. We were able to show that formal analysis based

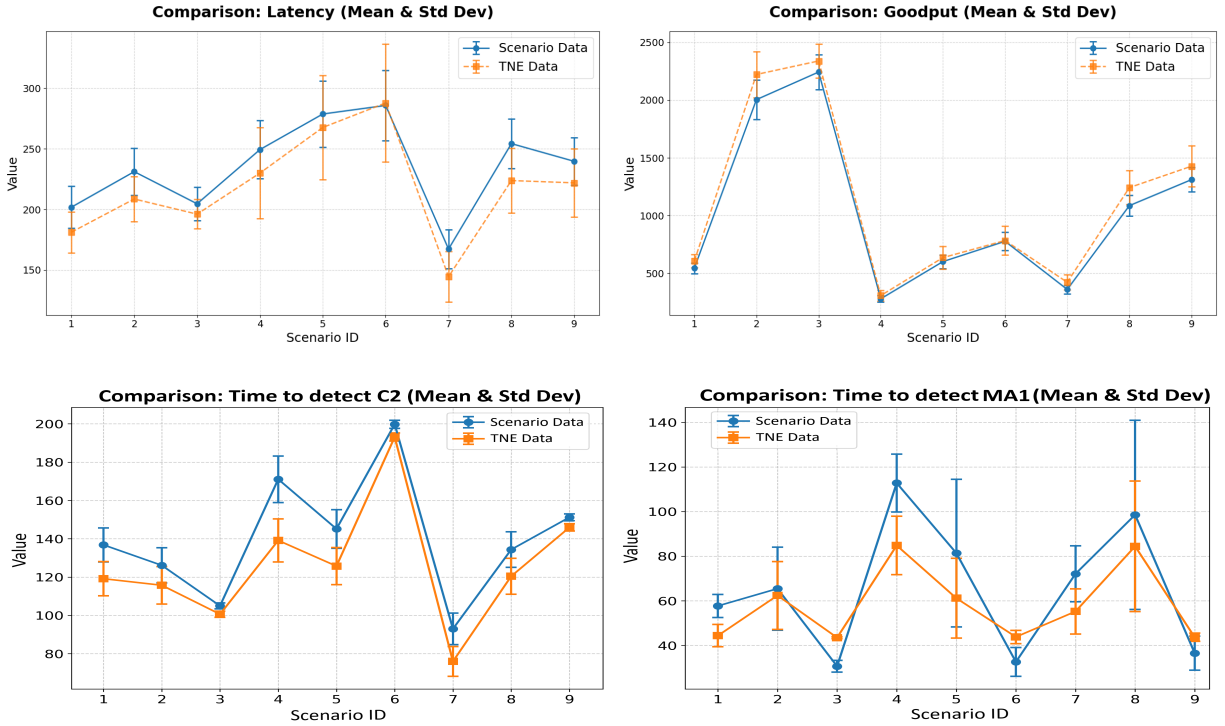


Figure 7: Semantic alignment verification using latency (top left), goodput (top right), Op duration (time-to-detect) monitoring DNS queries counts (bottom left), and Op duration monitoring DNS query rate (bottom right) metrics between Test and Evaluation (TNE) and Maude. Error bars show one standard deviation from the mean. Our HCS model comes close (around one standard deviation) to the statistics collected on the testbed.

on the Maude-HCS models replicated with substantial faithfulness experimental results in many cases; but it is not yet clear in general what it means for a formal model to align with such implementations.

An important next step is to *invert the process*: to first (i) design and analyze a new hidden communication system using the Maude-HCS framework, to formally analyze its performance and undetectability properties, and to improve and fine-tune the system’s design to meet desired performance and indistinguishability goals (which is orders of magnitude easier and more informative than evaluating, optimizing and fine-tuning an actual implementation); and then (ii) generate an implementation of the fine-tuned and thoroughly analyzed HCS protocol design from its formal model. For greater assurance, by using code generation methods similar to the one proposed in [34], it may even be possible to generate a high-quality implementation in an automatic and correct-by-construction way.

We developed the current models by carrying out by hand the theory transformations needed to specify and analyze hidden communication protocols, and to convert non-deterministic models into purely probabilistic models that can be used for SMC analysis by Monte Carlo simulation. An important next step is to implement the already existing algorithmic definitions of the different formal patterns in [44] used in the Maude-HCS modeling and analysis tool

chain: automating these transformations will make it much easier for communication protocol designers to use Maude-HCS in order to reach high-quality designs based on solid performance and security estimations of their behavior *before* their designs are implemented.

Simulating and analyzing performance involves considering scenarios with large numbers of actors, some engaged in normal communication, some carrying out hidden communication. For formal models and their SMC analysis to scale up to large scenarios, together with SMC parallelization methods such as those in [4], which are already supported by QMaude, we also need to develop *abstraction methods* that reduce the state space while still capturing all the system’s relevant features.

Acknowledgments

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No HR001125CE019.

References

- [1] Gul Agha. 1986. *Actors: a model of concurrent computation in distributed systems*. MIT press. <http://hdl.handle.net/1721.1/1692>
- [2] Gul Agha, José Meseguer, and Koushik Sen. 2006. PMAude: Rewrite-based specification language for probabilistic object systems. *Electronic Notes in Theoretical Computer Science* 169:1–20.

- Computer Science* 153, 2 (2006), 213–239. <https://doi.org/10.1016/j.entcs.2005.10.040>
- [3] Gul Agha and Karl Palmskog. 2018. A Survey of Statistical Model Checking. *ACM Trans. Model. Comput. Simul.* 28, 1 (2018), 6:1–6:39.
- [4] Musab AlTurki and José Meseguer. 2011. PVEStA: A parallel statistical model checking and quantitative analysis tool. In *International Conference on Algebra and Coalgebra in Computer Science*. Springer, 386–392. https://doi.org/10.1007/978-3-642-22944-2_28
- [5] Christel Baier, Luca de Alfaro, Vojtěch Forejt, and Marta Kwiatkowska. 2018. Model checking probabilistic systems. *Handbook of Model Checking* (2018), 963–999. https://doi.org/10.1007/978-3-319-10575-8_28
- [6] Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. 2021. SoK: Computer-aided cryptography. In *2021 IEEE symposium on security and privacy (SP)*. IEEE, 777–795.
- [7] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2017. Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams. *Proceedings on Privacy Enhancing Technologies* (2017).
- [8] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vitor Nunes. 2020. Poking a hole in the wall: Efficient censorship-resistant internet communications by parasitizing on webrtc. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 35–48.
- [9] Bruno Blanchet. 2012. Security protocol verification: Symbolic and computational models. In *International conference on principles of security and trust*. Springer, 3–29.
- [10] Carlos E Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. 2017. JANI: quantitative model and tool interaction. In *Tools and Algorithms for the Construction and Analysis of Systems: 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings, Part II 23*. Springer, 151–168. https://doi.org/10.1007/978-3-662-54580-5_9
- [11] Vincent Cheval and Itsaka Rakotonirina. 2023. Indistinguishability beyond difference equivalence in proverif. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*. IEEE, 184–199. <https://doi.org/10.1109/CSF57540.2023.00036>
- [12] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. 2007. All about Maude - A high-performance logical framework how to specify, program and verify systems in rewriting logic. In *All About Maude - A High-Performance Logical Framework - How to Specify, Program and Verify Systems in Rewriting Logic (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics))*. 1–819. <https://doi.org/10.1007/978-3-540-71999-1>
- [13] M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet, and C. Talcott. 2007. *All About Maude - A High-Performance Logical Framework*. Springer LNCS Vol. 4350.
- [14] Manuel Clavel, Francisco Durán, Steven Eker, Santiago Escobar, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, Rubén Rubio, and Carolyn Talcott. 2024. Maude manual (version 3.5). *SRI International University of Illinois at Urbana-Champaign* (Sept. 2024). https://doi.org/wiki/Maude_Manual_and_Examples
- [15] Christopher Connolly, Patrick Lincoln, Ian Mason, and Vinod Yegneswaran. 2014. TRIST: Circumventing Censorship with Transcoding-Resistant Image Steganography. In *4th USENIX Workshop on Free and Open Communications on the Internet (FOCI 14)*. USENIX Association, San Diego, CA. <https://www.usenix.org/conference/foci14/workshop-program/presentation/connolly>
- [16] Véronique Cortier, Steve Kremer, and Bogdan Warinschi. 2011. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning* 46, 3 (2011), 225–259.
- [17] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. 2015. Uppaal SMC tutorial. *International journal on software tools for technology transfer* 17 (2015), 397–415. <https://doi.org/10.1007/s10009-014-0361-y>
- [18] Defense Advanced Research Projects Agency. 2018. *RACE: Resilient Anonymous Communication for Everyone*. <https://www.darpa.mil/research/programs/resilient-anonymous-communication> Accessed: 2026-02-25.
- [19] Defense Advanced Research Projects Agency. 2025. *PWND²: Provably Weird Network Deployment and Detection*. <https://www.darpa.mil/research/programs/provably-weird-network> Accessed: 2026-02-25.
- [20] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-generation Onion Router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13 (SSYM'04)*. USENIX Association, Berkeley, CA, USA, 21–21. <http://dl.acm.org/citation.cfm?id=1251375.1251396>
- [21] Zach Dorfman and Jenna McLaughlin. 2018. Botched CIA Communications System Helped Blow Cover of Chinese Agents. *Foreign Policy* (15 August 2018). <https://foreignpolicy.com/2018/08/15/botched-cia-communications-system-helped-blow-cover-chinese-agents-intelligence/> Accessed: 2026-02-25.
- [22] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. 2013. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 61–72.
- [23] Kevin P Dyer, Scott E Coull, and Thomas Shrimpton. 2015. Marionette: A programmable network traffic obfuscation system. In *24th USENIX Security Symposium (USENIX Security 15)*. 367–382.
- [24] Erik Ekman and Björn Andersson. n.d. *iodine: Official git repo for iodine dns tunnel*. <https://github.com/yarrick/iodine> Accessed: 2026-02-27.
- [25] Daniel Ellard, Christine Jones, Victoria Manfredi, W Timothy Strayer, Bishal Thapa, Megan Van Welie, and Alden Jackson. 2015. Rebound: Decoy routing on asymmetric routes via error messages. In *2015 IEEE 40th Conference on Local Computer Networks (LCN)*. IEEE, 91–99.
- [26] fadoss. n.d. *umaudemc: Unified Maude model-checking tool*. <https://github.com/fadoss/umaudemc> Accessed: 2026-02-26.
- [27] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies* (2015).
- [28] Daniel Galán, Víctor García, Santiago Escobar, Catherine Meadows, and José Meseguer. 2023. Protocol Dialects as Formal Patterns. In *European Symposium on Research in Computer Security*. Springer, 42–61. https://doi.org/10.1007/978-3-031-51476-0_3
- [29] Arnd Hartmanns and Holger Hermanns. 2014. The Modest Toolset: An integrated environment for quantitative modelling and verification. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 593–598. https://doi.org/10.1007/978-3-642-54862-8_51
- [30] Josh Karlin, Daniel Ellard, Alden W Jackson, Christine E Jones, Greg Lauer, David P Mankins, and W Timothy Strayer. 2011. Decoy routing: Toward unblockable internet communication. In *USENIX workshop on free and open communications on the Internet (FOCI 11)*.
- [31] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14–20, 2011. Proceedings 23*. Springer, 585–591. https://doi.org/10.1007/978-3-642-22110-1_47
- [32] Si Liu, Huayi Duan, Lukas Heimes, Marco Bearzi, Jodok Vieli, David Basin, and Adrian Perrig. 2023. A Formal Framework for End-to-End DNS Resolution. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 932–949. <https://doi.org/doi/10.1145/3603269.3604870>
- [33] Si Liu, José Meseguer, Peter Csaba Ölveczky, Min Zhang, and David A. Basin. 2022. Bridging the semantic gap between qualitative and quantitative models of distributed systems. *Proc. ACM Program. Lang.* 6, OOPSLA2 (2022), 315–344.
- [34] Si Liu, Atul Sandur, Jos'e Meseguer, Peter Csaba Ölveczky, and Qi Wang. 2020. Generating Correct-by-Construction Distributed Implementations from Formal Maude Designs. In *Proc. NASA Formal Methods: 12th International Symposium, NFM 2020, (Lecture Notes in Computer Science, Vol. 12229)*. Springer, 22–40.
- [35] Mastodon contributors. 2016. *Mastodon: Your self-hosted, globally interconnected microblogging community*. <https://github.com/mastodon/mastodon> Accessed: 2026-02-28.
- [36] Richard McPherson, Amir Houmansadr, and Vitaly Shmatikov. 2016. CovertCast. *Proceedings on Privacy Enhancing Technologies* (2016).
- [37] Catherine Meadows. 2022. Predicting Asymptotic Behavior of Network Covert Channels: Experimental Results. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*. 2–3.
- [38] Braham Lotfi Mediouni, Ayoub Nouri, Marius Bozga, Mahieddine Dellabani, Axel Legay, and Saddek Bensalem. 2018. Bip 2.0: Statistical model checking stochastic real-time systems. In *International symposium on automated technology for verification and analysis*. Springer, 536–542. https://doi.org/10.1007/978-3-030-01090-4_33
- [39] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN prover for the symbolic analysis of security protocols. In *International conference on computer aided verification*. Springer, 696–701.
- [40] José Meseguer. 1992. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science* 96, 1 (1992), 73–155.
- [41] José Meseguer. 1993. A Logical Theory of Concurrent Objects and its realization in the Maude Language. In *Research Directions in Concurrent Object-Oriented Programming*, Gul Agha, Peter Wegner, and Akinori Yonezawa (Eds.). MIT Press, 314–390.
- [42] José Meseguer. 2014. Taming distributed system complexity through formal patterns. *Science of Computer Programming* 83 (2014), 3–34. <https://doi.org/10.1016/j.scico.2013.07.004>
- [43] José Meseguer. 2026. Programming and Verifying Actor Systems in Rewriting Logic. In *Concurrent Programming, Open Systems and Formal Methods - Essays Dedicated to Gul Agha to Celebrate His Scientific Career (Lecture Notes in Computer Science, Vol. 16120)*, José Meseguer, Carlos A. Varela, and Nalini Venkatasubramanian (Eds.). Springer, 446–474. https://doi.org/10.1007/978-3-032-05291-9_19
- [44] J. Meseguer and P.C. Ölveczky. 2026. The Timed P Transformation for Distributed Real-Time Systems. (2026). Submitted for publication.
- [45] Paul Mockapetris. 1987. *Domain Names - Concepts and Facilities*. RFC 1034. Internet Engineering Task Force (IETF). <https://www.ietf.org/rfc/rfc1034.txt>
- [46] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 97–108.

- [47] Milad Nasr, Hadi Zolfaghari, and Amir Houmansadr. 2017. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2037–2052.
- [48] Boel Nelson, Elena Pagnin, and Aslan Askarov. 2024. Metadata privacy beyond tunneling for instant messaging. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 697–723.
- [49] P. C. Ölveczky and J. Meseguer. 2007. Semantics and Pragmatics of Real-Time Maude. *Higher-Order and Symbolic Computation* 20, 1–2 (2007), 161–196.
- [50] Vern Paxson. 1999. Bro: a system for detecting network intruders in real-time. *Computer Networks* 31, 23–24 (1999), 2435–2463.
- [51] Vitor Pereira, Ahmed Irfan, Vinod Yegneswaran, Nick Feamster, Prateek Mittal, and Vitaly Shmatikov. 2025. Position Paper: A Case for Machine-Checked Verification of Circumvention Systems. *Free and Open Communications on the Internet* (2025).
- [52] Marc B Rosen, James Parker, and Alex J Malozemoff. 2021. Balboa: Bobbing and weaving around network censorship. In *30th USENIX Security Symposium (USENIX Security 21)*. 3399–3413.
- [53] Rubén Rubio, Narciso Martí-Oliet, Isabel Pita, and Alberto Verdejo. 2023. QMaude: Quantitative Specification and Verification in Rewriting Logic. In *Formal Methods - 25th International Symposium, FM 2023, Lübeck, Germany, March 6–10, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 14000)*, Marsha Chechik, Joost-Pieter Katoen, and Martin Leucker (Eds.). Springer, 240–259.
- [54] Sonia Santiago, Santiago Escobar, Catherine Meadows, and José Meseguer. 2014. A formal definition of protocol indistinguishability and its verification using Maude-NPA. In *International Workshop on Security and Trust Management*. Springer, 162–177. https://doi.org/10.1007/978-3-319-11851-2_11
- [55] Zhen Sun and Vitaly Shmatikov. 2023. Telepath: A minecraft-based covert communication system. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2223–2237.
- [56] The Zeek Project. 2025. *The Zeek Network Security Monitor*. <https://zeek.org/> Accessed: 2026-02-28.
- [57] Two Six Tech RACE Project. n.d.. *destini*. <https://github.com/tst-race/destini> Accessed: 2026-02-27.
- [58] Two Six Tech RACE Project. n.d.. *raceboat*. <https://github.com/tst-race/raceboat> Accessed: 2026-02-27.
- [59] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J Alex Halderman, Nikita Borisov, and Eric Wustrow. 2020. Running refraction networking for real. *Proceedings on Privacy Enhancing Technologies* (2020).
- [60] Ryan Wails, Rob Jansen, Aaron Johnson, and Micah Sherr. 2023. Proteus: Programmable protocols for censorship circumvention. *Free and Open Communications on the Internet* (2023).
- [61] Ryan Wails, Andrew Stange, Eliana Troper, Aylin Caliskan, Roger Dingleline, Rob Jansen, and Micah Sherr. 2022. Learning to behave: Improving covert channel security with behavior-based designs. *Proceedings on Privacy Enhancing Technologies* (2022).
- [62] Philipp Winter, Tobias Pulls, and Juergen Fuss. 2013. ScrambleSuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. 213–224.
- [63] Yawning Angel. n.d.. *obfs4 (The obfuscator) Protocol Specification*. <https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt> Accessed: 2026-02-25.
- [64] Hadi Zolfaghari and Amir Houmansadr. 2016. Practical censorship evasion leveraging content delivery networks. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1715–1726.

A Modeling Hidden Communication in Maude-HCS

We illustrate the basic ideas for modeling and analyzing hidden communication in Maude-HCS with a simple round trip time protocol, RTT, and a transformed version, WRTT, that uses RTT to hide the transmission of a stream of bytes, for example a stream of sensor measurements. An RTT session has two actors A and B. Actor A periodically sends an RTT request to B containing a timestamp $ts(T)$, a digital representation of the current time. B responds to the request with a pair consisting of the timestamp received from A and the timestamp corresponding to the current reading of the clock by B. When A receives the response, the roundtrip time (the current clock reading minus the request timestamp) is computed

and added to the senders local list of round trip times. RTT is transformed to transmit hidden information from the B to the A. Using a formal pattern the actors A and B are wrapped in meta actors that do the embedding and extracing of hidden information, leaving A and B unchanged. We present the result of a further transformation (optimization) that merges the attributes of the base and meta actors and combines the effects of the base and meta actor rules. Thus, in the WRTT protocol B is modified to embed a byte of hidden information in the lower 8 bits of response timestamp, and A is modified to extract that hidden byte in addition to computing the round trip time, which, as we shall see, is indeed the *correct* round trip time in spite of B’s modified timestamp.

The conjecture is that the hidden communication with WRTT is difficult to detect since it doesn’t change the message size or timing of RTT, and the lower order bits of actual timestamps are likely to be unpredictable due to clock reading randomness. We can study this conjecture using Maude-HCS, by specifying the RTT protocol and its transformation to WRTT, and further specifying what can be observed and how the adversary might use what is observed.

The Maude-HCS model of the RTT protocol is a PMaude model of the form $\mathcal{R}_{rtt}$, which is the result of applying the timed P transformation from [44] to the RTT protocol $\mathcal{R}_{rtt}$ by choosing probability distributions Π for message delays. RTT itself is modeled as a real-time rewrite theory $\mathcal{R}_{rtt}$ that provides RTT’s Real-Time Maude (RTM) semantics[49] and includes definitions of the attributes of the two actors, the rules defining the actors’ behavior. $\mathcal{R}_{rtt}$ is then modularly extended with a specification of adversary capabilities, and and specification of analysis scenarios.

The specification of $\mathcal{R}_{rtt}$ uses an abstract notion of time (sort Time) which can be instantiated to natural numbers, non-negative rationals, or floating point numbers for different analyses. Floating point approximation to real numbers is used for statistical analysis.

In the following we explain the Maude rules for $\mathcal{R}_{rtt}$ and show how $\mathcal{R}_{rtt}$ is transformed into the WRTT protocol $\mathcal{R}_{wrtt}$. We then describe how observables are collected to support analysis by statistica model checking and auditing.¹¹

The periodic behavior of A is specified using Real Time Maude (RTM) timers. A timer is represented by a term of the form $[T, P, data]$ where T has sort Time (the time remaining), P has sort Nat (the period), and data is auxiliary data, not used in the RTT protocol.

There are two rules for the behavior of the RTT sender A and one rule for the receiver B.

The sender rule [sndRttReq] specifies sending an RTT request $rttReq(ts)$ when the timer reaches zero.

```
cr1 [sndRttReq] :
  < A : Snd | timers: [tt ; per ; data], tsq: ts1,
    clock: T, stopT: Tstop, rcv: B, atts >
  =>
  < A : Snd | timers: [tt0 ; per ; data], tsq: (ts1 ; ts),
    clock: T, stopT: Tstop, rcv: B, atts >
  [ (to B from A : rttReq(ts)) ]
  if tt == zero
  /\ tt0 := (if T ge Tstop then infly else nat2t(per) fi)
  /\ ts := t2ts(T) .
```

The term $t2ts(T)$ represents the timestamp obtained by reading a clock at time T. The sender A updates its timestamp queue attribute

¹¹The specification and analysis scenarios are available in the Maude-HCS repository.

tsq: by appending ts to the list of previously sent timestamps ts1. This allows A to connect each response from B with a specific request from A. A also resets its timer either to *infty* (the infinite time) if the clock value T is greater than the stop time, Tstop, or to the request period, per, (converted from a natural number to a time, nat2t(per), otherwise.) Setting the timer value to *infty* is equivalent to turning the timer off, as decrementing *infty* by a finite value remains *infty*. In this case the sender will stop sending.

The term [(to B from A : rttReq(ts))] is an RTM message scheduling task. Such tasks are rewritten to delayed messages dly(M, dT) by RTM rules. Delayed messages remain in the configuration until the amount of time dT has elapsed.

The sender rule [rcvRttResp] specifies sender A's behavior when a response rttResp(ts, ts0) arrives from the receiver B. The computed round trip time, computeRtt(T, ts), is appended to the attribute rttq:. A then waits for the timer to reach zero to send the next rtt request. The passage of time is specified by the RTM tick rule that computes the minimal time that must elapse before some action (i.e., some rule firing) is enabled.

```
rl[rcvRttResp]:
  < A : Snd | clock: T, rcv: B, timers: [tt0 ; per ; data],
    tsq: (tsl0 ; ts ; tsl1), rttq: rttl , atts >
  (to A from B : rttResp(ts,ts0))
=>
  < A : Snd | clock: T, rcv: B, timers: [tt0 ; per ; data],
    tsq: (tsl0 ; tsl1),
    rttq: (rttl ; computeRtt(T,ts)), atts > .
```

The rule [rcvRttReq] specifies that when a request rttReq(ts) from A is received by the receiver B, then B sends a response message of the form rttResp(ts, t2ts(T)) to A containing the received timestamp and its current timestamp. B does not change its state.

```
rl[rcvRttReq]:
  < B : Rcv | clock: T, snd: A, atts >
  (to B from A : rttReq(ts))
=>
  < B : Rcv | clock: T, snd: A, atts >
  [ (to A from B : rttResp(ts,t2ts(T))) ] .
```

Guided by the formal pattern for WRTT, rules from the RTT protocol $\mathcal{R}_{rtt}$ are transformed into those of $\mathcal{R}_{wrtt}$ to carry hidden information as follows. The sender A and the receiver B share a common attribute ctr: j, with j initially 0, that they increment each round. This makes possible for A and B to generate a shared sequence of (pseudo) random numbers.

The sender rule [sndWrttReq] is unchanged. The receiver rule is transformed: B now generates a random byte, denoted by

$b(\text{sampleUniWithIntX}(i, 256))$

and embeds it into its timestamp using a function embed(ts, j, byte). The new rule also increments the counter.

```
cr1[rcvWrttReq]:
  < B : bWRcv | clock: T, snd: A, bctr: i, ctr: j,
    byteList: bytel, atts >
  (to B from A : rttReq(ts0))
=>
  < B : bWRcv | clock: T, snd: A, bctr: (s i), ctr: (s j),
    byteList: (bytel ; byte), atts >
  [(to A from B : rttResp(ts0,ts1))]
```

```
if ts := t2ts(T)
/\ byte := b(sampleUniWithIntX(i,256))
/\ ts1 := embed(ts,j,byte) .
```

The sender rule to receive the RTT response is modified to extract the hidden information, add the result byte to its byteList: attribute, and increment its counter.

```
cr1[rcvWrttResp]:
  < A : bWSnd | clock: T, rcv: B, rttq: rttl, ctr: j,
    tsq: (tsl ; ts0 ; tsl1), byteList: bytel, atts >
  (to A from B : rttResp(ts0,ts1))
=>
  < A : bWSnd | clock: T, rcv: B,
    rttq: (rttl ; computeRtt(T,ts0)), ctr: s j,
    tsq: (tsl ; ts0 ; tsl1),
    byteList: (bytel ; byte), atts >
  if byte := extract(tsl1,j) .
```

The embedding function replaces the low order 8 bits of the timestamp (viewed in binary representation) by the result of xoring the byte with the low order 8 bits of the random number given by the counter value, j. The extracting function xors the low order 8 bits of the timestamp with the same random number (using the value of the synchronized counter). Since this the same random number used by the receiver, the embed byte is recovered.

Once we have specified the real-time rewrite theories $\mathcal{R}_{rtt}$ and $\mathcal{R}_{wrtt}$ we can use the timed *P* transformation in [44] (choosing the same Π for both protocols), as well as the *Sim* and *M* transformations described in Section 2, to perform SMC analysis of RTT and WRTT by means of the executable PMAude theories $M(\text{Sim}(\mathcal{R}_{rtt}))$ and $M(\text{Sim}(\mathcal{R}_{wrtt}))$, or, as explained below, of their extensions when an *observer* is added to the state.

Configurations consisting of a sender A and receiver B can be used to generate sample execution traces that can be inspected or subject statistical model checking, for example to determine the expected roundtrip time over a given time interval.

To analyze properties based on observables and measure differences between the RTT and WRTT scenarios, we add an observer actor to the configuration. The observer attributes are chosen to model information an observer/adversary can obtain under different conditions. To study the effect of the hidden communication, we define an observer that collects the list of responses that the sender actor receives. Such an observer has the form

$\langle \text{obs} : \text{Observer} \mid \text{sent: timeMsgList} \rangle$

The observation mechanism is specified as a set of functions executed by the RTM rules. The RTM rule [sendDelayed] for converting scheduling tasks [M] to delayed messages dly(M, T) also executes the function sendTimeAction(conf0, M, T) where conf0 is the rest of the configuration. The rule [msgArrival] that rewrites a message with zero delay, dly(M, zero), to a receivable message M also executes the function arrivalTimeAction(conf, M). The RTM rule [infty] applies when the minimal time to advance is *infty*, indicating the end of the execution. This rule applies the finalAction to the configuration and terminates the execution. The finalAction function may collect additional information from the actors and may add summary information to the observer. By default, all these functions are the identity on configuration. Each system analysis should define them as needed.

In the RTT/WRTT examples, `arrivalTimeAction` is defined to add the pair (T, M) to the observer `sent`: attribute, thus collecting all the responses received with the time of receipt. Thus, when the stop time is reached, the observer object will have the form:

`< obs : Observer | sent: (T1,M2) ; ... ; (Tk,Mk) >`

where $M1 \dots Mk$ are the responses received by the sender.

In the RTT/WRTT cases, the `finalAction` function adds three new attributes to the observer. The value of the `enList` attribute is a list of the entropies of the 8 low order bits receiver timestamps (the second timestamp in the response pair) in the `sent`: attribute. The value of the `enAv`: attribute is the average of the values in the entropy list. The value of the `rttAv`: is the average of the round trip times recorded by the sender in the `rttq`: attribute.

For example, the results below are from eight successive runs of the PMaude models of RTT and WRTT with parameters: `startTime 999.`, `stopTime 10999.`, `Period 500`, and link delay between sender and receiver sampled from a normal distribution with mean 50.0 and standard deviation 10.0.

RTT

```
enAv: 6.0863390095915237e-1, rttAv: 9.7904761904761898e+1
enAv: 7.06767431818905e-1, rttAv: 9.8571428571428569e+1
enAv: 6.5459027078376264e-1, rttAv: 9.8904761904761898e+1
enAv: 5.9602399389404914e-1, rttAv: 9.8619047619047606e+1
enAv: 7.0761801320459572e-1, rttAv: 9.7809523809523796e+1
enAv: 6.2943397982393068e-1, rttAv: 9.804999999999997e+1
enAv: 7.8639975258512451e-1, rttAv: 9.729999999999997e+1
enAv: 5.463652286124957e-1, rttAv: 9.794999999999989e+1
```

WRTT

```
enAv: 7.2691476093154683e-1, rttAv: 9.7904761904761898e+1
enAv: 7.4344054910605573e-1, rttAv: 9.8571428571428569e+1
enAv: 7.2093973227398722e-1, rttAv: 9.8904761904761898e+1
enAv: 6.981128141131786e-1, rttAv: 9.8619047619047606e+1
enAv: 7.2158497284097312e-1, rttAv: 9.7809523809523796e+1
enAv: 7.1022824513931959e-1, rttAv: 9.804999999999997e+1
enAv: 7.1032439283023907e-1, rttAv: 9.729999999999997e+1
enAv: 7.0959202890719641e-1, rttAv: 9.794999999999989e+1
```

Informally it appears that the round trip time averages have similar distributions in the two cases, while the entropy seems to be slightly higher in the WRTT case; but a thorough application of SMC is still needed to confirm this impression.

For detectability we can compute the expected value of the entropy at some high confidence for each case. We could also pick a threshold, for example .7, and carry out true/false positive analysis.

B Expressing Mixed Properties

It is intuitive to express the operating duration quantitative property as a mixed properties in QuaTEX. To specify the operating duration (time of detection) for the C2 detector,

```
OpDurationC2() =
if
(s.rval("ToDNQueryPostNAT(C,NTHRESHOLD)") == 0.0)
then
discard
else
(s.rval("ToDNQueryPostNAT(C,NTHRESHOLD) -
ExfilStartTime(getMonitor(C))"))
fi;
```

```
eval E[OpDurationC2()];
```

The `ToDNQueryPostNAT(C, NTHRESHOLD)` computes the total number of post-NAT DNS queries in the adversary's observation (which is included in the configuration C), scanning the trace from the beginning, and at the point in time T when the threshold `NTHRESHOLD` is exceeded it returns the difference between T and the time when Alice started exfiltrating its files (computed by `ExfilStartTime` over the monitor object which is also in the configuration).

This is a *conditional expectation*, which focuses the measure on sample runs where the alarm actually fires. Sample runs where the threshold is not exceeded are discarded. `QMaude` reports the expected value with its confidence radius, and the number of simulation runs that were discarded. This property is precise, and the measure is statistically grounded.